Recommending tasks in online judges

Recommending Tasks in Online Judges

Giorgio Audrito[A,C], Tania Di Mascio[B], Paolo Fantozzi[D], Luigi Laura[C,D], Gemma Martini[C], Umberto Nanni[D], and Marco Temperini[D]

[A] University of Torino, [B] University of L'Aquila, [C] Italian Association for Informatics and Automatic Calculus (AICA), [D] Sapienza University of Rome
Italy

Abstract. Online Judges are e-learning tools used to improve the programming skills, tipically for programming contests such as International Olympiads in Informatics and ACM International Collegiate Programming Contest.
In this context, due to the nowadays broad list of programming tasks available in Online Judges, it is crucial to help the learner by recommending a challenging but not unsolvable task. So far, in the literature, few authors focused on Recommender Systems (RSs) for Online Judges; in this paper we discuss some peculiarities of this problem, that prevent the use of standard RSs, and address a first building brick: the assessment of (relative) tasks hardness.
We also present the results of a preliminary experimental evaluation of our approach, that proved to be effective against the available dataset, consisting in all the submissions made in the Italian National Online Judge, used to train students for the Italian Olympiads in Informatics.

Keywords: recommender systems, programming contests, e-learning

1 Introduction

In recent years we have witnessed the diffusion of Programming Contests (PCs), i.e. competitions in which participants are faced a set of tasks that require writing computer programs. The importance and the effectiveness of PCs in the process of learning computer programming and, more generally, computer science has been broadly emphasized [4, 3, 5, 8, 12, 15].

Training for PCs relies heavily on Online Judges (OJs), also called Programming Online Judges, that are web based e-learning tools where a learner can submit solutions to a programming task. The learner chooses a task and reads its statement; then, online or offline, he writes a solution that is submitted to the OJ, that verifies the correctness, usually by testing it against a certain number of test cases, and the efficiency, by checking that the running time and/or the memory usage is under some limit.

However, as observed in [17], the large number of tasks available to users is a typical example of *information overloading scenario*: an unexperienced user has to choose from thousands programming tasks, many of which are probably too difficult for him. Just to provide some examples, University of Valladolid

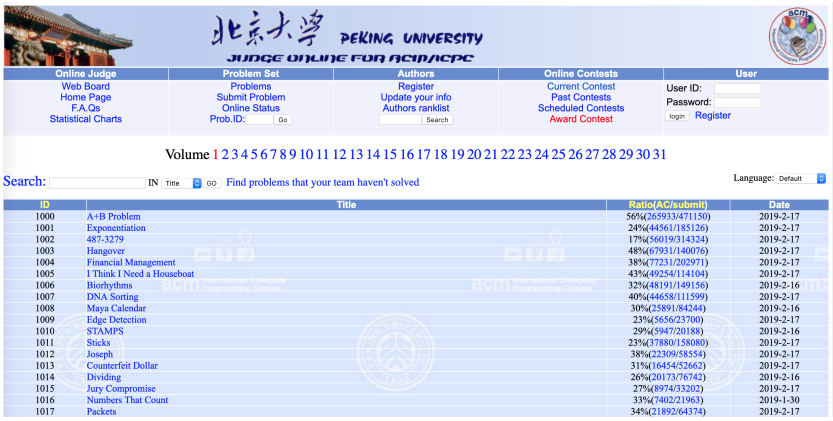

Fig. 1. The list of available problems in the Peking University OJ.

Online Judge has more than 200k users and 2k tasks, whilst SPOJ accounts approximately 600k users and 6k (public) tasks. In Figure 1 is shown the typical interface with the list of tasks in an OJ platform, whilst in Figure 2 is shown an example of a programming task.

With such numbers, a Recommender System (RS) for the users is definitely needed, to help them finding the next task. Traditional RS approaches can be very broadly divided into two categories: Content Based ones, in which the recommendations derive from features of the items to be suggested, and Collaborative Filtering approaches, in which the suggestion is based on the items chosen by users *similar to the current one*.

There are, however, some peculiarities of Online Judges that prevent the use of a general Recommender System:

- the user slowly improves his abilities, one task after the other, so the general concept of user *preferences* does not apply: recommending a movie or a book differs significantly from recommending a task; a user will probably still like a movie after one year, whilst he might find a task too easy after the same amount of time.
- Users with *similar* skills, i.e. users to whom we might want to suggest the same set of tasks, might behave very differently in OJs, thus preventing us from considering them *similar*. For example, one might solve all the tasks involving a given skill, while the other might just solve one task, related to that skill, and then move on to tasks involving different skills.

Note that the above issues are typical of RSs in e-learning tools, thus successful approaches in this field might be extended to more general cases of TEL systems.

In this paper we propose an algorithmic approach to a building block of a task recommender system: given a list of tasks solved by the users, we want to

OIS2019 – Round 3

Online, December 19th, 2018

paths • EN

Railway Schedule (paths)

The railway network in the *Pordenone* county consists of N train stations connected by $N - 1$ tracks (X_i, Y_i) so that from every station is possible to reach any other station: in other words, the tracks form a *tree*.
This choice makes the transportation system extremely inefficient: trains going in opposite directions cannot cross each other on a single track, so they need to perform lengthy and complex manoeuvres to pass each other. The new administration founded its campaign trail on changing this situation once and for all... and now it's time to keep promises!
Edoardo, the local leading expert in logistics, already has a mind-blowing idea for fixing the situation: making each track one-way, so that no crossings will ever occur! Of course, the tricky part is choosing the orientations so that the service remains acceptable for the majority of the population. After inspecting the traffic patterns, Edoardo discovered that most people travel between one of M pairs (A_i, B_i) of stations. Thus, an orientation of the tracks will be considered *acceptable* by the population only if for each such pair, either a path from A_i to B_i or a path from B_i to A_i should exist. However, many acceptable orientations exist and Edoardo cannot choose among them, otherwise his system would be deemed as unfair: the only solution is to use *all* of them in a periodic schedule of daily track orientations. Help Edoardo design such a schedule by counting how many acceptable orientations exist! Since this number may be large, report it modulo 1 000 000 007.

Fig. 2. An example of a problem from a programming contest; this task is taken from the final contest of the 2019 edition of the Italian Team Olympiads in Informatics (OIS) [2].

estimate relative hardness of the tasks, i.e. finding a ranking of the tasks from the easiest to the hardest.

Notice that the number of users who solved a given task, which could be seen as a proxy for the hardness of the task, is not a good indicator: popular hard tasks might have more users that solved them compared to easy, unpopular, tasks.

Our approach is based on the construction of a graph, where the nodes are the tasks and the (weighted) directed edges represent the number of users that solved one task before the other. We tested the effectiveness of our approach on the data from the OJ used by the secondary school students training for the Italian Olympiads in Informatics (Olimpiadi Italiane di Informatica - OII) [10], and the preliminary experimental results confirm the effectiveness of our approach.

This paper is organized as follows: the next section provides the necessary background related to programming contests, online judges, and recommender

systems, whilst our approach is detailed in Section 3. In Section 4 we describe our experimental findings and concluding remarks are addressed in Section 5.

2 Background

In this section we provide the reader the necessary background concerning programming contests, online judges, and recommender systems.

2.1 Programming Contests

A programming contest is a competition in which contestants are faced with a set of programming tasks, also called problems, to be solved in a limited amount of time and/or with a limited amount of memory usage.

A single task can be broken into different subtasks of increasing complexity: basic techniques might be enough to solve, within the given time and/or space limits, some of the subtasks whilst the most difficult ones might require very specific algorithmic techniques and data structures.

We mention some popular programming contests:

- The International Olympiads in Informatics (IOI), that are an annual programming competition for secondary school students patronized by UNESCO. `http://www.ioinformatics.org/`
- The ACM International Collegiate Programming Contest (ICPC) is a multitier, team-based, programming competition operating under the auspices of ACM. `https://icpc.baylor.edu/`
- The very recent International Olympiads in Informatics in Team (IOIT), that started in 2017, that are a team competition, like ACM ICPC, differently from IOI (individual competition). Currently there are only four nations involved: Italy, Romania, Russia, and Sweden. `https://ioi.team/`
- Google Code Jam, that is based on multiple online rounds that concludes in the World Finals. `https://code.google.com/codejam/`.
- Facebook Hacker Cup, that is *an annual worldwide programming competition where hackers compete against each other for fame, fortune, glory and a shot at the coveted Hacker Cup*. `https://www.facebook.com/hackercup/`

2.2 Online Judges

The Online Judges are, usually, web based platforms that provide a large number of programming tasks to be solved. There are several popular OJ platform, we cite the already mentioned University of Valladolid Online Judge `https://uva.onlinejudge.org`, Sphere Online Judge (SPOJ) `https://www.spoj.com/`, CodeChef `https://www.codechef.com/`, and Peking University Online Judge `http://poj.org`.

In the literature, the first reference to Online Judge dates to the paper of Kurnia, Lim, and Cheang [13]. A brief survey on OJs can be found in [17], whilst more extended surveys on tools and techniques for automatic evaluation of solutions submitted to OJs can be found in [1, 6].

2.3 Recommender Systems in OJs

As already observed in the introduction, despite the large amount of literature devoted to RS, the peculiarities of recommendation in OJs, where the relation user-item is way more complex than the typical RS cases, prevent from using standard techniques and forces the development of ad-hoc methods.

However, so far few research focused in the recommendation of tasks in OJs: in [14] the authors use the traditional collaborative filtering method with a new similarity measure adapted to the case, whilst in [17] is presented an approach based on fuzzy logic, refining a previous approach [16]. In [7], Caro and Jimenez tackled the problem by considering user-based and similarity-based approaches. An alternative approach is detailed in [11], where is defined a framework that can allow recommendations and that can foster motivation in students by means of a lightweight, badge-based, gamified approach.

Our approach differs from the ones cited above because we aim at solving a subproblem: can we derive the ranking of the tasks, ordered by their hardness?

3 Ranking Tasks in Online Judges

In this section we details our approach. Our goal is to provide a rank of the tasks, based only on the submissions made by the users. Thus, our input data contain all the submissions to an OJ platform. Our approach is based on the following assumptions:

- a task is solved by a user if and only if he has obtained the maximum possible score on it
- having two tasks t_1 and t_2 where t_1 is harder than t_2, then each user, most of the times, will solve them in order of ascending difficulty (so first t_2 and the t_1)
- it is possible to estimate the difficulty of a task, just using the users' submissions, without any knowledge about the users

The submissions are sorted by user and timestamp, to have the ordered sequence of the tasks for each user. Since we assume that each user solves the tasks in ascending order of difficulty, then we can consider this sequence as monotonically increasing in terms of difficulty.

So, considering d_i as the measure of the difficulty of t_i, we assume that in each sequence $d_i \leq d_j \iff i < j$. Now we build a weighted directed graph with **tasks as nodes** and where each edge in the graph (t_i, t_j, w) means that $d_i < d_j$ in w different sequences.

Since that it is possible that if a task could be solved by some users just after it has been uploaded on the platform, even if it is not harder than the last one solved by the user, then we try to reduce the error induced by this, avoiding counting the sequences where the timestamp of t_i is prior than the uploading of t_j on the platform in the weight of the edge (t_i, t_j).

Since that a sequence of length n will create $O(n^2)$ edges, we expect to have a much dense graph. Moreover, an old task (uploaded at the beginning of the

utilisation of the platform) will likely have more out edges than the newer tasks. To decrease the bias given by the age of the tasks, we create N random walks on the graph with random length, chosen in rl_m and rl_M.

Each random walk starts from a random node, and then, in each iteration a random out edge e_i of the node n is chosen such that the probability p of choosing e_i is equal to

$$p(e_i) = \frac{w_i}{\sum\limits_{j \mid e_j \in out(n)} w_j}$$

At this point we have N sequences such that:

$$\forall\, t_i, t_{i+1} \in S \implies (t_i, t_{i+1}) \in G$$

where S is the set of N random walks' paths, G is the original graph built from submissions. So we build a subgraph $G' = (V', E')$ of $G = (V, E)$ such that

$$\forall (s, d, w') \in E' \implies (s, d, w) \in E,\ w' = |\{p \mid (t_i, t_{i+1}) \in p,\ p \in S,\ t_i = s,\ t_{i+1} = d\}|$$

The resulting graph should have the same order of magnitude for all the edges. This means that the bias given by the age of the task is reduced drastically. Then, the edges are once again filtered out leaving only one direction. In practice we will maintain only the edge with the maximum weight between $s \to d$ and $d \to s$.

This final graph is used to define an order between nodes, using different metrics. For example a score for a node n:

$$m(n) = \frac{\sum\limits_{i \mid e_i \in in(n)} w_i}{\sum\limits_{j \mid e_j \in out(n)} w_j + \sum\limits_{i \mid e_i \in in(n)} w_i}$$

4 Experimental Evaluation

In order to evaluate the effectiveness of our approach, we considered a dataset with the data from the OJ used by the secondary school students training for the Italian Olympiads in Informatics (Olimpiadi Italiane di Informatica - OII) [10, 9].

In particular, this dataset had:

- 321430 submissions
- 366 tasks
- 3928 users

We considered only the submissions that solved the tasks, so we reduced to 68859 submissions, where the distributions of the users with respect to the tasks are shown in Figure 3. We computed 100000 random walks, that is a 10-factor over the number of edges of the starting graph. The resulting graph contains 366 nodes (i.e., one for each task in the OJ platform) and 121803 edges.

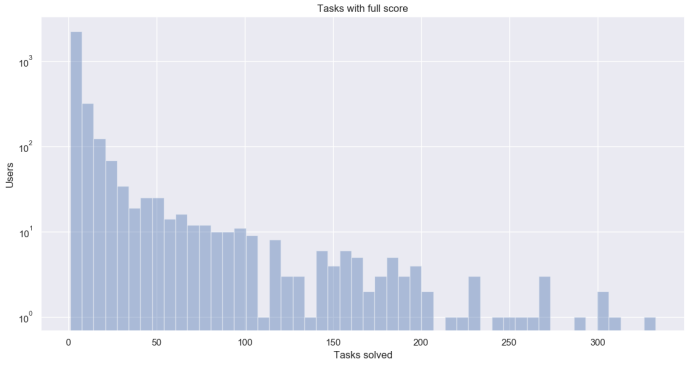

Fig. 3. Distribution of number of solved tasks per users

Judging the hardness of a task is, by definition, a subjective problem. In order to assess our results we only considered the top 25 tasks, and divided them into five buckets of five tasks each. We asked some three experts (i.e. the tutors that mantain the platform) to evaluate our results by sort the five buckets in the order of the hardness of the tasks included. Two experts sorted the buckets in the same order obtained by the algorithm, whilst the third one swapped the third and fourth bucket. We plan to test our approach on data from other OJs, but the validation of the results is a complex issue by itself.

5 Conclusions

In this paper we proposed a graph based approach to estimate the relative hardness of tasks in OJs. This is a basic building block of a recommending system to suggest the next task to be solved by a user.

We also performed an experimental evaluation of our approach against the data from the OJ used by the secondary school students training for the Italian Olympiads in Informatics (Olimpiadi Italiane di Informatica - OII) [10, 9].

Our preliminary results seem promising, and we plan to carry on our investigations by testing it with different data; furthermore, as mentioned in the previous section, the problem of the evaluation of the results is a complex task by itself, and we plan to try alternative approaches also in this directions, including a comparison with traditional recommendation methods (including random orders).

References

1. K. M. Ala-Mutka. A survey of automated assessment approaches for programming assignments. *Computer science education*, 15(2):83–102, 2005.

2. N. Amaroli, G. Audrito, and L. Laura. Fostering Informatics Education through Teams Olympiad. *Olympiads in Informatics*, 12:133–146, 2018.
3. O. Astrachan. Non-competitive programming contest problems as the basis for just-in-time teaching. In *Frontiers in Education, 2004. FIE 2004. 34th Annual*, pages T3H/20–T3H/24 Vol. 1, Oct 2004.
4. G. Audrito, G. B. Demo, and E. Giovannetti. The role of contests in changing informatics education: A local view. *Olympiads in Informatics*, 6, 2012.
5. M. Blumenstein, S. Green, S. Fogelman, A. Nguyen, and V. Muthukkumarasamy. Performance analysis of game: a generic automated marking environment. *Computers and Education*, 50:1203–1216, 2008.
6. J. Caiza and J. Del Alamo. Programming assignments automatic grading: Review of tools and implementations. In *INTED2013 Proceedings*, 7th International Technology, Education and Development Conference, pages 5691–5700. IATED, 4-5 March, 2013 2013.
7. M. Caro-Martinez and G. Jimenez-Diaz. Similar Users or Similar Items? Comparing Similarity-Based Approaches for Recommender Systems in Online Judges. In D. W. Aha and J. Lieber, editors, *Case-Based Reasoning Research and Development*, volume 10339, pages 92–107. Springer International Publishing, Cham, 2017.
8. V. Dagienė. Sustaining informatics education by contests. In *International Conference on Informatics in Secondary Schools-Evolution and Perspectives*, pages 1–12. Springer, 2010.
9. W. Di Luigi, P. Fantozzi, L. Laura, G. Martini, E. Morassutto, D. Ostuni, G. Piccardo, and L. Versari. Learning analytics in competitive programming training systems. In *2018 22nd International Conference Information Visualisation (IV)*, pages 321–325, July 2018.
10. W. Di Luigi, G. Farina, L. Laura, U. Nanni, M. Temperini, and L. Versari. oii-web: an interactive online programming contest training system. *Olympiads in Informatics*, 10:195–205, 2016.
11. T. Di Mascio, L. Laura, and M. Temperini. A framework for personalized competitive programming training. In *2018 17th International Conference on Information Technology Based Higher Education and Training (ITHET)*, pages 1–8, April 2018.
12. G. Garcia-Mateos and J. L. Fernandez-Aleman. Make learning fun with programming contests. In *Transactions on Edutainment II*, pages 246–257. Springer, 2009.
13. A. Kurnia, A. Lim, and B. Cheang. Online judge. *Computers & Education*, 36(4):299–315, 2001.
14. R. Y. Toledo and Y. C. Mota. An e-learning collaborative filtering approach to suggest problems to solve in programming online judges. *Int. J. Distance Educ. Technol.*, 12(2):51–65, Apr. 2014.
15. T. Wang, P. Su, X.and Ma, Y. Wang, and K. Wang. Ability-training-oriented automated assessment in introductory programming course. *Computers and Education*, 56:220–226, 20011.
16. R. Yera and L. Martínez. A recommendation approach for programming online judges supported by data preprocessing techniques. *Applied Intelligence*, 47(2):277–290, Sep 2017.
17. R. Yera Toledo, Y. Caballero Mota, and L. Martínez. A Recommender System for Programming Online Judges Using Fuzzy Information Modeling. *Informatics*, 5(2):17, 2018.